



- (1) We expect IP-TCP use to be dropped in favor of ISO protocols by late 1989.
- (2) Vendor support expected after initial development/beta test period.
- (3) Periodic review and evaluation of new vendors/workstations for use in the AIM community.
- (4) Initial exploration of concepts followed by extensive development, testing, and dissemination.
- (5) Convert IMAP-1 to IMAP-2 -- see below.
- (6) Use of InterLisp is expected to phase out by late 1988 in favor of Xerox CommonLisp -- see below.
- (7) We expect use of the DEC 2060 to phase out between the 4th and 5th years.
- (8) This work will involve an extended period of development, testing, and dissemination, the exact schedule of which cannot be predetermined because it will depend on the results of other developments and experiments in this area.
- (9) There will be an ongoing involvement with the vendor to test and support new system developments.
- (10) We will use a small group of Stanford and AIM AI researchers and students to guide development and testing of distributed subsystems throughout the research period. Initially these will come mainly from the Stanford community which is easily accessible and already familiar with workstation use.
- (11) We will need to have enough of the systems R&D done to begin testing tools in a broader Stanford user community.
- (12) Dissemination to the general national AIM community will actually start earlier in an experimental mode (see 10) but full scale testing will begin in the 3rd year to prepare for a complete migration over the final 2 years.

Figure 1: Core System Development Schedule, Concluded

implementation. We had our X client well underway, when we discovered that TI and MIT decided to jointly develop a Common Lisp X-client (CLX) instead of a server.

Thus, in order to conserve our limited development resources, we chose to take a "wait and see" attitude, and redirect our programming efforts to the more immediate need of a distributed mail system. The time spent on our X-client was not wasted, since we gained a very deep insight into the protocol and its implementation, and planned to continue this project at a later date. In retrospect our choice was the correct one. CLX and a supporting CLX library for the creation of windows, menus, scroll-bars and other graphical objects is done, as is the implementation of a CLX-server which is due for alpha-release this June. Finally, The Common Lisp User Environment (CLUE) is now available from TI and is a window system on top of the CLX library which uses the Common Lisp Object System (CLOS). Thus, MIT and TI have provided a sufficient set of graphics primitives upon which one can build AI and systems tools in the X paradigm on TI Explorers. And, any other remote system which runs an X client and server will be able to have full duplex graphics communication with an Explorer. Currently, such a capability is available on Symbolics, SUNs, and VAXs.

Also, since CLUE is a more general window-system/user-environment on top of the CLX protocol primitives, it satisfies the long standing need for a portable window system for developing Common Lisp AI applications and will certainly find uses here at SUMEX-AIM.

Yet, much more work needs to be done in this area to fully develop and integrate remote graphics capabilities into Lisp machine systems in order to make low bandwidth connections a reality. We alluded to this topic in last year's report, and as yet, no one has begun to work in this area.

In the coming year we want to insure that cross-country connections will indeed support remote graphics and give usable response time. Success of this work will mean that one can use LISP machine systems from TELENET, ARPANET, or an EtherTIP connection throughout the SUMEX-AIM community.

4.3 - Remote Graphics Applications

Our emphasis in the area of remote access/graphics has evolved from proving the concept of remote windows and remote tools to building real systems on top of remote access capabilities for routine use.

TALK

We've added a new protocol and new service to the TALK (intra-machine user communication tool which employs both text and graphics) system we described previously. We added the 'Sketch' service type as well as the ability to communicate using the IP/TCP protocols over the Ethernet.

Several groups in the national Xerox Lisp machine community who used the TALK program we distributed requested the TCP/IP capability. In addition to implementing the TCP/IP layer, since most BSD 4.* UNIX systems have a TALK facility, we wanted to make the TTY mode of our TALK compatible with the UNIX implementation. This would have allowed us to TALK to users on any VAX or SUN; we actually got to the point (using shortcuts for testing) where we could TALK from a VAX to a Xerox Lisp machine. Unfortunately, we ran into a problem in that the packets that SUN UNIX generates for TALK are different than those generated by a VAX due to differences in word alignment. This is true of UNIX running on other architectures as well as the

UNIX TALK program failed to make all of its data types network independent. We are, however, able to use all the TALK modes (TEdit, Sketch and TTY) to communicate between any Xerox Lisp machines available via the ARPA/Internet and have done a trans-continental TALK with a user at MIT.

We implemented the new 'Sketch' service for TALK which uses an object-based graphics editor as the basis for communication instead of a WYSIWYG text editor. The addition of this service was mostly a test to see if TALK was properly layered so that services could be added without modifying the TALK program itself, as we had designed. For the most part this was the case, the 'Sketch' service was brought up without modifying TALK in any way. Later we made some minor modifications based on what we learned to further simplify the addition of new services.

To put TALK into routine use in the KSL, we included it (and a few other Courier-based servers) in our default Lisp image so that it is always available to users in order to increase the use of sophisticated multi-machine tools.

MacWorkstation

We have started a project based on the MacWorkstation software licensed by APPLE. The MacWorkstation program is designed to allow the Macintosh to start up programs on mainframes and receive graphic and text output to the MAC in a seamless fashion. We plan to write a Common Lisp window package (based on Common Windows) that uses MacWorkstation so we can connect (via Ethernet, Applenet or RS-232C/modem) to any of our Common Lisp engines and run the same piece of code on any and/or all of them. The MacWorkstation program uses a high level protocol which handles windows, menus, files, events, text and graphics and provides higher level access to the Macintosh graphic facilities than the native system library routines.

We do not plan to remote the existing window package of any given system; this interface will be written without regard to the window system on the machine on which it is being developed. We plan on providing a new library of window commands based on the emerging Common Windows standard. This will both simplify the design of the system and allow existing code that conforms to the CommonWindows standard to be used with little or no modification. This interface code can then be used on all Common Lisp engines (even mainframes) in our environment, including TI Explorers, SUN workstations, Symbolics and Xerox Lisp machines.

A Macintosh II is being used to run MacWorkstation and a Xerox Lisp machine is being used for the Common Lisp development. The two are currently tied together using a serial line which will eventually be upgraded to an TCP/IP Ethernet connection. We expect the resulting interface to work with the full range of Macintosh systems including the smaller ones typically in home use. It is possible to extend the capabilities of the MacWorkstation software on the Macintosh side, so once the basic system is in place we can then add whatever features we feel are necessary (if anything) to handle real Lisp applications.

We have already identified a group in our own laboratory that will be able to use the finished product. The SightPlan system needs to interface to BB1 output on an Explorer so that a user can see graphically how the site is being laid out. That graphics system is being implemented on a Macintosh since they can not use the Explorer screen which is full of BB1 data. We imagine systems based on the Common Windows/MacWorkstation interface that require the symbolic processing power of a dedicated lisp engine but also need to be available to a large number of

users. We have test software running now which allows the Lisp machine to open windows on the Macintosh and do some simple graphics.

Other Remote Tools

We finished a tool (MONITOR) that allows a Xerox Lisp machine to examine the display of another workstation. This tool shows a shrunken version of the entire remote machine's display in one window and allows you to examine a smaller portion of the display in full size in another. We plan to use this to remotely examine the display on the Oncology clinic machine (which runs the ONCOCIN expert system) when one of the physicians calls up with a problem. This tool is another built on top of the Courier server we described in previous years.

5 - File Access and Management

A stable, efficient mechanism for storing and organizing data is central to any computing environment, and is one of the most challenging issues in the move to distributed, workstation-based computing. It is necessary to provide standard services, such as file backup, archival, a flexible, intuitive naming facility, and data interchange services (e.g., software distribution). Also, as the amount of data being manipulated grows, it will become more and more important to have powerful tools for managing hierarchies of files. We plan to support the community with a number of UNIX-based file servers, like the VAX-based servers (see Figure 7) and the SUN-based server (see Figure 5) in use at SUMEX for several years. These will require continued SUMEX-AIM development, however. By keeping the number of servers small, the distributed namespace problem should be manageable in the near term. Current UNIX file servers are relatively cheap and fast. UNIX has many of the needed facilities, e.g., backup, long names, hierarchical directory structure, some file property attributes, data conversion, and limited archival tools.

However, while general issues of networking, remote memory paging services, and flexible file access have received considerable attention in both the academic and commercial development of file servers, there seems to be only slow development of other critical operational needs. For instance, the much-used file archiving system of the DEC 2060 (sometimes called off-line cataloged storage) has no analog service in the UNIX systems. Perhaps this is the result of UNIX having its origin in the small computer world where the number of users and volume of data has traditionally been quite low. Our efforts are going into trying to adapt a commercial system developed by UniTech to allow users to manage large file systems by providing access to and from off-line tape storage as well as maintain a historical archive of files. We have had a well structured organization for managing disk usage and accounting on our 2060 system and we plan to duplicate some of these features on the SUN-4 such that we can manage its usage in much the same way. A UNIX accounting system will be put in place to allow cost accounting and provide a quota enforcement capability on the distributed file server(s).

UNIX has always been weak in the management of large-scale file backup and with the advent of disks of near gigabyte size, it is clear that an organized approach to their usage is essential. In support of the long-term goals of the distributed community, we have been reviewing more advanced data storage methods. Optical disk systems are attractive but have not progressed as quickly as many had predicted. However, it seems likely that this sort of equipment would be ideally suited to satisfying our requirements for the archiving of files. Helical-scan magnetic tape equipment might also serve this function well. However, in both cases the absence of industry standards introduces some level of risk when planning to use these types of data storage equipment. We plan to continue our investigations of these technologies this coming year and to make a decision.

The AppleTalk/UNIX File Service package (AUFS) developed at Columbia University provides file support on a UNIX server so that Macintosh workstations can connect and see representations of the files present in the same icon-based format as the file system on the native Macintosh. File transfers are invoked by moving icons around the Macintosh screen just as if the UNIX server were an extension to the internal Mac disk. AUFS functions will be expanded to provide better and more service to our large Macintosh community.

6 - Electronic Mail

Electronic mail continues as a primary means of communication for the widely spread SUMEX-AIM community. As reported last year, the advent of workstations has forced a significant rethinking of the mechanisms employed to manage such mail in order to ensure reliable access, to make user addressing understandable and manageable, and to facilitate keeping the mail software distributed to workstations as simple, stable, and maintainable as possible. We are following a strategy of having a shared *mail server* machine which handles *mail transactions* with *mail clients* running on individual user workstations. The mail server can be used from clients at arbitrary locations, allowing users to read mail across campus, town, or country.

The mail server acts as an interface among *users*, *data storage*, and *other mailers*. Users employ a *mail access protocol* (MAP) to retrieve messages, access and change properties of messages, manage mailboxes, and send mail. This protocol should be simple enough to implement on relatively inexpensive machines so that mail can be easily read remotely. This is distinct from some previous approaches since the mail access protocol is used for *all* message manipulations, insulating the user client from all knowledge of how the mail is actually stored on the server. This means the the mail server can utilize whatever data storage and access methods are most efficient to organize the mail on a particular server system.

The first prototype Interim Mail Access Protocol (IMAP) was designed with this in mind. As noted in our previous report, we developed a prototype IMAP server on the DEC-20 and client on Xerox D-machines. The resulting mail environment proved to be quite usable and some D-machine users were able to use it as an alternative to the DEC-20 mail environment in their daily mail work.

During implementation of the prototype client we observed that the protocol had several deficiencies which made certain mail concepts difficult or impossible to implement. For example, there was no way to notify the client of newly arrived mail and inadequate extensibility in the protocol made it difficult to add such a functionality. Furthermore, it was a "lock-step" protocol with no mechanism for multiplexed operation, leading to synchronization problems. Finally, we rethought parts of our design based on our examination of several related projects, including MIT's PCMAIL and various POP2-based systems.

Our second-generation Interactive Mail Access Protocol (IMAP2) addresses these concerns. Instead of the lock-step query/response model of IMAP, IMAP2 uses tagged commands and data and explicitly allows unsolicited data to be sent from the server to the client. IMAP2 introduced a more formal structure to server-to-client path; in particular, all data is now identified unambiguously. IMAP2 also addressed various performance issues, such as allowing the fetching of multiple types of data items simultaneously. Furthermore, IMAP2 introduced server-based message searching and the concept of a structured "envelope" representing the information stored in the header of an RFC822 message.

An IMAP2 server manipulates the actual file store copy of the user's incoming electronic mail under direction from the IMAP2 client. As noted above, the client has

no knowledge of the (possibly operating system dependent) format of mail on the server's file store; the IMAP2 protocol provides its own representation of mail and the server translates between this and its host system file store conventions.

The IMAP2 client issues a series of *fetch* commands to retrieve data from the server. A fetch command has two arguments: a *message sequence* and the name of the data item to be fetched. A message sequence can be a single message index, a range of message indices, or a list of numbers or ranges. A message index is one of a set of consecutive numbers which provide a quick pointer to that particular message in the mailbox. A mailbox with 20 messages uses indices 1 through 20.

As an example, a typical FETCH command might be

```
A0001 FETCH 2:7,10 ENVELOPE
```

meaning "fetch the envelope data for messages 2 through 7 and message 10". "A0001" is a random tag generated by the client, and used by the server in the completion response to an IMAP2 command. An envelope is in a format very much like a Lisp S-expression, and represents, in a structured fashion, data such as the From/To/cc/bcc lists, message Subject, Message-ID, etc.

There is presently no structured representation for the text of the message itself; the client must fetch the "RFC822" data attribute which is returned as a text string of the message in traditional RFC822 format. This is due to our immediate need for a system operationally usable in today's environment and also our feeling that the requirements for such a structured representation are not well-enough understood at this time.

There are other message attributes which can be fetched, the most interesting of which is the "FLAGS" attribute which contains various status flags associated with the message. The operation of removing a message from the mailbox consists of setting the system "\DELETED" flag for a message via the "STORE" command, and subsequently doing an "EXPUNGE" command.

Other operations in IMAP2 include LOGIN/LOGOUT for user authentication, SELECT for mailbox access, STORE to update/alter attributes associated with a message, EXPUNGE to permanently remove deleted messages from the mailbox, and SEARCH to do remote searches for messages based on various attributes.

The SEARCH command is quite sophisticated; for example the command

```
A0021 SEARCH FROM "SMITH" SUBJECT "REPORT"  
        SINCE "20-MAR-88"
```

requests the server to return a list of message indices of all messages from SMITH with a Subject that includes the string "REPORT" that were placed in the mailbox after March 20, 1988. An elaborate set of remote search criteria have already been defined and implemented in the IMAP2 servers.

The server on the DEC-20 and Xerox Lisp client were upgraded to IMAP2. We have also implemented an IMAP2 server for UNIX, and have demonstrated a prototype client in Common Lisp for the Texas Instruments Explorer. The main thrust of our effort in the coming year, however, will be a client for the Macintosh.

6.1 - Xerox Lisp client

An IMAP2-based client program for the Xerox Lisp environment was useable by mid-year. However, it was not in wide-spread use and lacked the necessary final polishing and debugging needed for release to a large community of users. A major

push was made to make the necessary changes and the program was shuttled back and forth between members of the programming staff, who are specialists in different areas, in order to make the program as well-rounded as possible. Once the improved versions started to become available, the user community was steadily increased and where possible the resulting feedback was immediately reflected the next release. Along with the polishing of the client program, the documentation was completely revised and extended and the servers programs were further debugged.

Some of the specific changes to the Xerox Lisp client included the fixing of storage leaks, code optimization, basic window management and the splitting of the various command menus into several smaller menus and submenus, based on functionality. The major changes included the addition of the text compression and breaking features of the 'WEDIT' program (described in earlier reports) to the composition window to allow the mailer to freely convert between unbroken, streams of text and fixed length lines. Another major change was the addition of 'Zoom' mode for pulling together selected messages in the mail header browsing window. This allowed users to take full advantage of both the existing and extended text search capabilities that were added to the program without paying a penalty for retrieving excess mail headers from the server or doing manual searches via scrolling. A feature to search for mail files and allow the names of those files to be used as input to other commands with the click of a button was also added.

The Xerox Lisp client was further integrated into the KSL environment by adding it to one of the standard Lisp images (TCPFULL.SYSOUT), making it readily available to a larger community of users. The Xerox Lisp client is not going to be the mail client that the entire laboratory uses in the long term. The final push to finish the program was done in order to make available to a larger portion of the KSL a mail system based on the client and server model that everyone will be using eventually on other hardware. We wanted experience in running such a system and to allow users to have early feedback in the design of the ultimate system. The Xerox Lisp client will be maintained only as is reasonable as the mail system evolves so it can be used as a fast prototyping system for new ideas. The InterLisp-based IMAP2 layer of this client may be replaced by the Common Lisp one under development in order to reduce the maintenance costs of the various systems.

The Xerox Lisp client is currently the only fully functional, active client in our distributed mail system but probably will remain so for only a short while longer. Soon, we plan to make this client available to the Xerox Lisp user community along with the DEC-20 and UNIX IMAP2 servers.

6.2 - Texas Instruments Explorer client

We are now writing an IMAP2 client for the TI Explorer system using Common Lisp. Although this project is not yet completed, it already can read mail, set message flags, move/copy messages between mailboxes and much more. Still to do is the message composing and sending. Except for the window system which is Explorer dependent, we expect much of this code can be exported to other Common Lisp workstations allowing us to develop mail systems for these workstations too.

6.3 - DEC-20 IMAP2 server

The DEC-20 IMAP2 server was the first implementation of IMAP2 (and its predecessor, IMAP), and many of the ideas of our distributed mail environment were explored there first. This was due at least in part to the familiarity of the programmer with that traditional environment. The last major functionality addition to the DEC-20 server was the addition of the SEARCH facility last year. The current

version of the DEC-20 server is a complete and robust implementation of IMAP2 and we do not anticipate any further work on it.

6.4 - UNIX IMAP2 server

In our distributed mail system we expect to rely heavily on UNIX engines as servers. To this end we have written a UNIX IMAP2 Server (UIMS) for remote mail access. UIMS handles the full complement of IMAP2 commands and has been in alpha-test mode for the past month.

On a UNIX server a mailbox is represented by two files. The first is the actual text file containing the mail, and the second is an index file that caches a description of the text file's format. The index file is used to minimize UIMS disk i/o by keeping enough information about the text file and each message in that file to, first, speed random access to the text file, and second, make file access unnecessary for many of the IMAP2 protocol commands. For example, a binary representation of a message's date, flags and keywords is cached for each message. Thus queries on the state of these fields require no disk file access since a copy of the index file is kept in memory while a mailbox is selected. The index file also allows UIMS to implement a demand-paging algorithm to manage the core memory used for storing the text of each message as it is read or searched. As a consequence, only messages in which a user shows an interest need be kept in memory. In most instances this is only a small percentage of the number of messages in the text file. Such an algorithm conserves the server's resources, and is especially important under the UNIX operating system since it does not implement shared memory pages between processes.

For compatibility with MM-C, the Columbia University UNIX MM program (see below), UIMS and MM-C use the same text file for mail. MM-C has no knowledge of the index file associated with this file, and thus, UIMS will recreate the index file if the mail text file has been modified since UIMS last read it. Also, a locking mechanism is used to prevent simultaneous writes to the same mail text file. Unfortunately, locking under UNIX is not implemented in the operating system itself, and thus can be violated if a program not respecting file system locks attempts to write to a file that is open and locked by a program that does. All of the programs we have written and intend to write will respect these locks, as does MM-C.

UIMS also parses the MM-C init file, that describes a user's mail reading configuration, when this file exists. UIMS and MM-C have proved to be very successful when used in tandem for remote, and local mail reading, respectively.

6.5 - Transition Strategy and Plan

While in transition from our mainframe-oriented paradigm to a distributed computing environment, the ability to read mail locally, from home or from geographically distant locations is clearly necessary at all points in this process. To make this move as easy as possible we hoped to be able to find a mail program that ran under UNIX to be used as a replacement for MM-20, the mail program now used at SUMEX-AIM by the majority of our community members. We were fortunate to discover MM-C, an MM-20 clone written at Columbia University in C to run on UNIX-based systems. We became an alpha test site for MM-C, and brought it up on both a VAX 11/750 and a SUN-3/180. It has been used extensively by members of our staff, and is purported to be an excellent mail program. By June of this year we should have installed MM-C on our new SUN-4 server.

As local community members move from the DEC-20 to the SUN-4, they will initially

read their mail in the same mode as they did on the DEC-20, using MM-C if they do not have an Explorer or Xerox D-Machine. In the latter instances, they will use their respective clients in conjunction with the UNIX IMAP2 server discussed in the previous section. We anticipate that the SUN-4 will be used as a time-shared system for reading mail for the short term while the Mac client is being completed. When this is done, all such mail reading will be accomplished using the IMAP2 server/client paradigm.

In order to read mail from home one will dial-in to one of our EtherTIPs, TELNET to the appropriate server and run MM-C. The UNIX IMAP2 server is designed to be compatible with MM-C so that one can read their mail both locally and remotely without conflict.

It is also important to mention that Columbia University distributes MM-C with the statement "Permission is granted to any individual or institution to use, copy, or redistribute this software so long as it is not sold for profit, provided this copyright notice is retained". This makes MM-C an ideal mail reading program for the SUMEX-AIM community as we move away from the DEC-20, and towards our distributed environment, since SUMEX-AIM can freely distribute all of its mail-related software to its national community.

7 - System Building Tools

7.1 - Lisp System Performance

One of the key issues in selecting the systems for our distributed computing environment was the performance of Common Lisp. In order to assist us in evaluating the performance of Lisp systems, we undertook an informal survey of Common Lisp environments using two KSL AI software packages, SOAR and BB1. The data collection for this evaluation is close to complete but we have only been able to do preliminary data analysis. The results are presented in Appendix B.

In this survey we have focused on execution speed for simulated system runs and for system compilation. However, we emphasize that execution speed benchmarks are only one aspect of the system performance evaluation, especially for Lisp systems. Other crucial issues like programming and usage environments, compatibility with other systems, ability to handle "large" problems, and cost must also be considered.

Both SOAR and BB1 were chosen because they are implemented in pure Common Lisp, making them extremely portable. SOAR is a heuristic search based general problem solving architecture developed by Paul Rosenbloom and BB1 is a blackboard problem solving architecture developed by Barbara Hayes-Roth. Neither of these systems is an intensive user of numeric computation. These systems were initially developed in environments other than those tested and no attempt was made to optimize their performance for any of these tests.

The workstation systems to be tested were chosen based on their availability as well as projected applicability in AIM community environments. Since we were interested in "real world" results, we ran the tests on each machine in what seemed to be its standard operating mode. The machines tested include:

- Apple Mac II
- DEC (uVAX II and III)
- HP 9000/350
- IBM (Compaq 386 "PS/2 model 80" and RT/APC)
- SUN (3/75, 3/60, 3/260, 4/260, 4/280)

- Symbolics 3645
- TI (Exp 1, Exp 2, uExp)
- Xerox 1186

and the Lisp systems running on these machines include:

- Explorer Lisp 3.0+ (4.0)
- Franz Extended Common Lisp 2.0
- HP Common Lisp 1.0
- Kyoto Common Lisp
- Lucid Common Lisp 2.x
- Symbolics Lisp 6.1
- VAXLisp
- Xerox Common Lisp (Lyric)

Analysis of the data that were collected is still underway but several conclusions are evident at a high level:

- The fastest machines for running BB1 (the microprogrammed TI Explorer 2 and microExplorer Lisp machines) are not the same machines that run SOAR fastest (SUN-4 RISC machines).
- Within a factor of two of the best performance, a considerable range of workstations based on stock microprocessor chips as well as specially microprogrammed Lisp chips have comparable performance.

Thus, while the "jury has been out" on the issue of special microprogramming versus stock instruction sets for Lisp systems, it now appears that stock systems have closed the performance gap. Unless substantial performance improvements are forthcoming from the specially microprogrammed architectures, it appears that Lisp-based research can just as well be done in "standard" workstation environments which would ease many issues of program export and integration.

7.2 - Lisp Programming Environments

Even though performance gaps between microprogrammed Lisp systems and stock workstation implementations are narrowing, there still remains a significant difference in the quality of the development environments. The power of the development tools and environment is what has been the primary strength of Lisp machines, allowing rapid design, implementation, and debugging of complex programs. We believe the key to good development tools is integration, both in terms of consistency of interface, and in the ability to move seamlessly from tool to tool, carrying along appropriate data and state information. These qualities must be manifest in any KSL research computing system.

Over the years, KSL and AIM community AI systems have been implemented predominantly in the InterLisp, MACLisp, ZetaLisp, and more recently, Common Lisp dialects. Beginning in the early 1980's, our work moved from mainframe Lisp environments to workstation environments for many reasons, principally involving powerful tools for system development and debugging and graphical interfaces. Commercial versions of these tools, that evolved over many years in the Xerox D-Machine, Symbolics 36xx, LMI, and TI Explorer systems, have become an indispensable part of our work environment. Newer Lisp systems for workstations not specifically developed for Lisp have lacked many important features of these environments. Thus, in light of the runtime performance advances of stock workstations, we have attempted to summarize the key features of the Lisp machine

environments that would be needed in stock machine implementations in order to make them attractive in a development setting.

The full version of this draft specification can be found in Appendix C. The following summarizes some of the key points about our effort to define the central environmental issues. We are continuing to seek comments from others in the AIM and broader AI communities as well as work with vendors and sponsors to try to get these feature integrated into commercially available Lisp systems.

- These requirements represent a snapshot of the tools and technology available on today's machines. AI has historically and will continue to ride the crest of the wave of new computing technologies for the foreseeable future, which enable ever more complex systems. Thus, these are not static requirements and we expect to be able to take advantage of the future improvements in hardware, graphics, and software as they are generated by computer science research and industry.
- We have tried to sort out the key features of current systems that are important to our research work. Except where explicitly stated, everything in the write-up (Appendix C) describes this "core" of functionality. Some items are clearly more important than others, but all represent needs that really guide our decisions about which new systems can be broadly used in the KSL.
- We have two overriding goals in adopting future computing environments (which may seem to be or actually are in some conflict). We want the most powerful development environments we can get to facilitate the building of complex AI programs. But at the same time, we want to be able to share (import and export) research results and tools with colleagues in other labs and so must maximize the portability of code among systems. We believe that these goals can be approached jointly by the establishment and careful adherence to standards where possible, while continuing systems development where necessary.

The requirements discussion is organized according to a "layered" view of Lisp environments, beginning with the upper levels. This organization is a conceptual framework within which to describe the various parts of the environment but may not correspond in full detail to the way system modules are actually organized. The elements of our description include:

- Program Development Tools and Environment
 - Editor
 - Debugger
 - Inspector
 - Software Management Tools
 - Performance Monitoring and Analysis
 - Lisp Listener
- Languages and Utilities
 - Windowing
 - Multiple Processes
 - Common Lisp
 - Compilation
 - Input/Output

- Utilities
- Interface Toolkit
- Help System
- Status Information
- Printing
- Pretty Printing
- Lower Level Issues
 - Address Space
 - Memory Management
 - Dedicated Versus Shared Systems
 - Hardware Capabilities
 - Overall System Integration in the KSL

7.3 - General System Building Environment

Traditionally, a large set of languages and programming environments has been supported on the 2060 in order to encourage experimentation and development. We now believe that the experience gained in those years of broad experimentation can be distilled into a fairly small set of languages and tools, relieving the researcher of the need to learn many programming languages, while still providing the needed facilities to allow the experimentation to move further into the higher levels of knowledge representation systems and problem solving architectures. As we move to the workstation-based environment, we plan to phase out support for many of the languages we have offered in the past and concentrate on the most relevant languages for AI research and applications: C, Apple PASCAL, FORTRAN, InterLisp-D, ZetaLisp, and Common Lisp. Common Lisp has already achieved popularity as a standard (see page 74), and many projects are already using it. We expect to press for further adoption of Common Lisp as a community symbolic computing standard, consistent with prior investments in large software systems such as those which exist for on-going AIM projects. In addition, we will support important higher-level knowledge representation and problem solving architectures (e.g., S.1, KEE, Strobe, and others) as appropriate for community research and dissemination activities.

8 - Text Editing -- TMAX

One of the biggest obstacles in reducing our dependence on the 2060 is large document preparation, including sectioning, cross-referencing, and flexible bibliographic referencing. At the KSL this sort of processing has been done using SCRIBE (a registered trademark of Unilogic Ltd.), TeX, or LaTeX. There have been over fifty Xerox workstations (1108/9s and 1186s) in the KSL and as part of one experiment to move users off of the 2060, we worked to develop a system called TMAX (TEdit Macros And eXtensions) with the equivalent functionality of SCRIBE on these machines. This work is now finished and the following reports on the results of the experiment. The system has proven quite powerful for complex documents -- we know of two Masters theses that have been written using TMAX. We have turned the TMAX system over to Xerox where it is now part of their standard working environment. The Xerox AIS group may use TMAX for their next set of InterLisp-D manual releases. Besides Xerox and Stanford, TMAX is in use at SRI International and Ford Aerospace.

The Xerox workstations come with a text editor called TEdit. TEdit is a WYSIWYG "text formatter" (which is part of what SCRIBE does) but it lacks the ability to do more complex operations. TMAX does not try to mimic SCRIBE; this would be a monumental task. Rather it extends TEdit by implementing some of the more

commonly used features of SCRIBE. Currently there are five main areas of TMAX; numbering, table of contents, endnotes (similar to footnotes), forward and backward referencing, and indexing. There are three important points about the TMAX experiment. First it is completely menu-driven. To invoke any feature, the user simply moves the mouse to the appropriate item in the TMAX menu and buttons it. Secondly TMAX never changes the user's text. All the features are merely additions to the text. These additions may appear to be strings but they are really structured objects that are treated as single characters. This means users can delete any feature they add in the same way they would delete any other character. Finally, all the inserted features are shaded gray so users can distinguish them from the normal text.

We have written a twelve page help file on TMAX that uses all of the TMAX features. This document is both a description and demonstration of how to use TMAX. Below is a brief description of each of the TMAX main features.

8.1 - Numbering

TMAX has an extremely flexible numbering facility that allows the user to number arbitrary objects such as chapters, sections, figures, examples, etc. The numbering scheme is completely up to the user. TMAX allows the user to create and edit a tree of "number nodes" where each node defines a level of numbering. The user also has the ability to define initial number values, the format of each number, text before and after the number (useful for headings), and the font of the number. To insert a number, the user simply buttons the appropriate node of the tree and TMAX figures out what the number should be and inserts it. Suppose for example the user is writing a book. The user could define the top node as Chapter with subnodes Section, Subsection, etc. Each time the user buttons Chapter in the number tree, the next monotonically increasing chapter number will be inserted in the document. It will also reset all Chapter's subnodes to their initial values. To insert a Section number, the user buttons Section in the tree and so on. The numbers can be ordinary Arabic numbers or upper and lowercase letters or Roman numerals. In the example above, the chapters could be "1.", the sections "1.A" and the subsections "1.A.i". Each value depends on the values of its immediately superior node in the tree. Finally, the numbering does not have to be inserted consecutively. If a user inserts a new number between other numbers, TMAX will automatically adjust all the numbers after the new one.

This numbering mechanism insures uniform numbering throughout the document. If at any time the user wishes to change the numbering format, he just edits the definition in the number tree and TMAX takes care of all the rest.

8.2 - Table of Contents

Closely related to numbering is the "Table of Contents" facility. When this item is buttoned, TMAX creates a file containing all the numbers it has inserted along with any text the user specified before and/or after the number followed by a dotted leader and the page each number appears on. The file is sorted by page number and the user has the ability to exclude certain numbers from the file.

8.3 - Endnotes

Endnotes are like footnotes in that they insert a superscript number but the text associated with the number is appended to the end of the document rather than the bottom of the page. When a user buttons the Endnote item, TMAX inserts the next monotonically increasing number and then prompts the user for the text string

associated this the number. Like numbers endnotes need not be inserted consecutively. If an endnote is inserted between other endnotes, TMAX automatically adjusts all the endnotes after the new one. Finally, there is no correspondence between numbering and endnotes numbers.

8.4 - Forward and Backward Referencing

Whenever a number or endnote is inserted, the user has the option of "tagging" it. At any point in the document, the user can reference this tag in one of two ways. The reference can be by the value of the number (e.g. "see chapter 5") or by the page the number is on (e.g. "see page 7"). Of course the user could reference the same number both ways and get something like "see chapter 5 on page 7". Whenever a tag is specified, it is added to a reference menu. If the user is referencing the same tag is several places, he can simply insert the reference from the menu rather than retyping the tag each time.

8.5 - Indexing

TMAX supports two styles of indexing; simple and extended. With simple indexing, the user just specifies the phrase to be indexed. With extended indexing the user can 1) specify the phrase that will appear in the index, 2) specify the font of this phrase, 3) have the phrase sorted by some other phrase, and 4) optionally exclude the page number from the index (useful for index headings). Whenever an index is first specified, it is added to an index menu. If the user is indexing the same phrase is several places, he can simply insert the index from the menu rather than retyping the phrase each time. At any time the user can create an index file. This files contains the alphabetically sorted indices followed by the page numbers on which they appear. Optionally the user can request "manual style" indexing in which the page numbers are replaced by selected numbers. In this case the index specifies the chapter, section, etc. in which the index appears.

These are only the main TMAX features. There are several other minor features such as the ability to insert the date and/or time in several different formats.

9 - Distributed Information Resources and Access

There are many user needs for getting information from and about the computing environment, ranging from help with command syntax to sophisticated database queries. A distributed computing environment adds new complexities in making such information accessible and also new requirements for information about the distributed environment itself. We are adapting the many workstation-specific information tools to include distributed environment information such as workstation and server availability, "Finger" information about user locations and system loads, network connectivity, and other information of interest to users in designing approaches to carrying out their research tasks. In addition, we will have to develop general systems tools for monitoring and debugging distributed system performance to identify workstation and network problems. Finally, we must adapt and develop distributed system tools for remote database queries and organize the diverse sources of information of interest to AIM community members to facilitate remote workstation access to community, project, and personal information that has traditionally existed in ad hoc files on mainframe systems.

The Macintosh HyperCard tool provides a very powerful environment in which to hierarchically organize and provide access to information in the form of text, graphics, pictures, and even sound. We have begun development of a "KSL stack" for HyperCard that will initially include descriptions of KSL and AIM personnel,

SUMEX-AIM projects, SUMEX-AIM computing systems, KSL building maps, the KSL bibliography, etc. While the current version of HyperCard is a good environment for a prototype distributed information access system, several developments are needed from Apple for full utility. These include remote network access to stacks, allowing multiple simultaneous readers of stacks, and more flexibility of how stack "cards" are presented on workstation screens.

On another front, in conjunction with the SUN file server we have been integrating, we have mounted an experimental database system for remote information access using the commercial UNIFY database product. Our goal is to make access to the database information possible from a distributed workstation environment through network query transactions, as opposed to asking the user to log into the database system as a separate job and type in queries directly. This will facilitate remote information access from within *programs*, including expert systems, where the information can be filtered, integrated with other information, and presented to the user. The system will provide multi-user, multi-database access capability; that is, several users will be able to have access to a single database at the same time, and a single user will be able to have access to several databases at the same time.

The initial implementation of the remote query system was done on a TI Explorer and then adapted to the Xerox D-machine Common Lisp environment. The query interface on the Lisp machine communicates with the Sun UNIFY database system via the Remote Procedure Call (RPC) mechanism which underlays the NFS remote file access system. The Explorer calls a server on the SUN and sends an SQL/DML query command as an argument to a remote query procedure, and receives the retrieved data and/or a message sent back from the server. SUN UNIFY can already manage multiple databases, so a client can have several databases open at the same time. The operations on the database are transaction-oriented, and therefore the concept of a database access session is applicable. The access functions currently implemented are *open a database*, *close a database*, *retrieve data from a database*, *insert records into a database*, *delete records from a database*, *update the database*, *lock a database*, and *unlock a database*.

10 - Distributed system operation and management

The primary requirements in this area are user accounting (including authorization and billing), data backup, resource allocations (including disk space, console time, printing access, CPU time, etc.), and maintenance of community data bases about users and projects. Our accounting needs are a function of system reporting and cost recovery requirements. The distributed environment presents additional problems for tracking resource usage and will require developing protocols for recording various kinds of usage in central data base logs and programs for analyzing and extracting appropriate reports and billing information. We are now involved in analyzing the kinds of resource usage that can be reasonably accounted for in a distributed environment (e.g., printing, file storage, network usage, console time, processor usage, server access), and investigating what facilities vendors have provided for keeping such accounts. Data backup is, of course, closely related to the filing issue. We continue to use and improve network based file backup for many of our file servers.

11 - Mainframe, Server, and Workstation System Environments

The various parts of the SUMEX-AIM computing environment require development and support of the operating systems that provide the interface between user software and the raw computing capacity. This includes the mainframe systems, network servers, and the workstation systems. Following are some highlights of recent system software environment developments.

11.1 - TOPS-20

Our long-term plan to phase out the 2060 mainframe system has continued as scheduled. Development efforts on the 2060 have ceased, except where needed to keep the machine operational in the evolving distributed environment. This involves considerable work in areas such as file system archiving, retrieval, and backups; periodic updating, checkout, and installation of new versions of system software; the regular maintenance and updating of system host and network tables; and monitoring of and recovery from system failures, both hardware and software. Over the past year, the main areas of activity include:

- *TOPS-20 Domain Name Resolver Software* -- The Internet community has been in the process of converting to a domain naming scheme, to replace the flat address space of the old exhaustive host tables prepared by the Network Information Center. This past year, we worked very closely with a sister group at MIT to integrate a version of the TOPS-20 Domain Name Resolver software with the TOPS-20 Mail System. In addition, several other network utilities, including TELNET and FTP were updated to use the Domain Resolver, rather than the old HOSTS.TXT table. Most other TOPS-20's are still running an ISI version of the DOMAIN resolver, which is inferior in several respects, so this represents a significant effort at SSRG to maintain network communication with the national Internet community.
- *Long-term access to SUMEX data files* -- The TOP-20 approach to providing references to off-line (archived) files presents substantial overhead to the active file system. Since SUMEX has always had an interest in preserving the history of the intellectual achievement resulting from our project, we are quite concerned with access to even very old data files. Thus we have prepared a staged scheme for retaining access to the files of former users. Though access is not as convenient to these files as it is to those archived by current users, there is, nonetheless, the ability to retrieve files of former users of our Tops-20 system and its predecessor, the PDP-10 based TENEX system.
- *Cost Center accounting* -- During the past year, the 2060 accounting programs were further updated to reflect the SUMEX Cost Center structure (see Page 121). Monthly reports have been promptly generated to reflect on-going usage. As part of the cost center management, a concerted effort has been made periodically to review all of the SUMEX accounts, and remove those that were no longer appropriate.

11.2 - UNIX

We run the Stanford (SULTRIX) version of the UNIX system that is distributed by the University of California at Berkeley on our shared VAX 11/750 file servers. SULTRIX adds support for Sun Micro Systems Network File Service (NFS), and the PARC Universal Packet (PUP) protocol that is necessary for our D-machines. The two VAX systems are used extensively as file servers, and minimally as time-sharing systems. One of these systems (ARDVAX) is used for systems development and the UNIX version of the IMAP server has been written and debugged on it. Little system development effort is done on these beyond staying current with operating system releases and useful SULTRIX community developments.

11.3 - Sun File Server

The SUN-3/180 file server (KNIFE), briefly mentioned in last year's report has been fully integrated into our distributed environment. In addition to providing file storage for about 50 users, it represents a prototype for the recently delivered SUN-4/280 in terms of implementation issues.

Imagen Host Software was installed to allow print spooling from the file server or any of its clients to any of the several Imagen Printers in the lab. This software was then updated to take advantage of the UltraScript (PostScript like document description language) on the Imagen 3320.

The InterLeaf Technical Publishing Software, and Frame Maker Software were installed for evaluation purposes.

The Unify Database Software, installed last year, continued to be used to administrative database applications.

The Columbia AppleTalk Package (CAP) was installed to allow local Macintoshes to access Apple-UNIX File Servers (AUFS) through Kinetics FastPath AppleTalk/Ethernet gateways.

A copy of the UNITECH UNIX file system backup/archival software was acquired for evaluation. The limited evaluation period ended before we could complete our evaluation, however, but we will probably try it again for our Sun-4/280 system, as an archival system is very important, and this package, although not perfect, looks like a good approach.

A 6250 BPI 1/2" magnetic tape Unit was added to the Sun-3/180 file server, and operational procedures put in place for regular file system backups.

11.4 - Xerox D-Machines

A considerable, but shrinking part of the SUMEX-AIM community continues to use Xerox Lisp; primarily on Dandetigers and Doves (Xerox 1109, 1186). A donation of hardware from an industrial source increased the number of Dorados (1132) from 1 to 2.

The Xerox workstations proved reliable and economical to repair, averaging under \$150 in repair costs per workstation for the year.

The *Lyric* release of the Xerox implementation of Common Lisp (which was beta-tested at SUMEX in March of '87 and saw general release in the Fall) proved so stable that it is still in use here a year later. The previous release, *Koto*, is still in use (by users of the commercial *KEE* and *Strobe* packages), but does not require or receive much support time from the staff. We will shortly beta-test the next release of Xerox Lisp--*Medley*.

We are also involved in the testing of a possible new hardware platform for the Xerox Lisp environment. The ONCOCIN system was used as a test program and the results of the experiments are encouraging. (A fringe benefit of the test is that research ONCOCIN has been upgraded from *Koto* to *Lyric* sooner than had been planned.)

The *Lyric* release required the reorganization of our font directories, containing several thousand files. (The font file names were changed to speed up lisp's search mechanism.) We deviated from the Xerox approach by further subdividing font directories according to font family name. The change required patching the lisp system software to exploit the new organization. This resulted in directories that are

much easier to maintain and increased speed in font searches -- in some situations nearly twenty times faster.

As with any new lisp release we had to update the code and documentation for our over twenty *Lispusers* packages. This effort ranged from simple recompilation to serious rewriting to account for changes in the system and to add new features based on accumulated user feedback. New *Lispusers* packages we made available to the national community this past year included TCP and UDP time packages with both client and server routines, a new device driver for a color film recorder, client implementations of the Sun RPC and NFS protocols, the *MONITOR* package for viewing remote displays, and the *SKETCHTALK* package for remote graphic interaction between workstation users.¹

We wrote an experimental Xerox Lisp *imagestream* driver for a Bell & Howell CDI-IV film recorder. This device was connected to a public 1108 via a 19.2 Kb serial line. In its HPGL (Hewlett Packard Graphics Language) emulator mode, the device was capable of drawing only single width straight lines in several colors. We used some bitmap-to-line drawing routines from our (previously implemented) plotter driver and a scheme using intermediate bitmaps (for the color planes) to produce a driver that could accurately render any image from the Xerox workstation on the film recorder (in color). This allows us to make photographic prints and slides of a fairly high quality. Patches to the Xerox *SKETCH* drawing program were necessary to make it work correctly in color mode. (These patches are applicable to other color devices.) The film recorder itself was donated by Bell & Howell and is on loan from the Computer Science Department.

We completed the Common Lisp implementation of Sun RPC (*Remote Procedure Call*) and NFS (*Network File System*) begun last Summer. The package was then used as part of a prototype system in which a database was accessible over the network to several types of workstation. Xerox has taken over responsibility for maintenance, improvement, and distribution of the package.

The Info-1100 discussion list which we sponsor continued to serve its readership on the ARPA Internet, Usenet, Bitnet, and CSNet. New subscriptions roughly equalled cancellations in number. Among the beneficiaries are other NIH-sponsored projects at Ohio State University and the University of Maryland.

The Xerox NS file server was reconfigured to run automatic nightly incremental backups from three of the T-300 disk drives to the file system on the fourth drive. Although this backup to disk does not give us the ability to retrieve files from the distant past (as tape backups do) it does give us the ability to restore a damaged file system to within a day of the damage. With three disk packs per drive and monthly full backups we are able to retrieve files from the previous 60 to 90 days. One problem with the previous system was automatic backup of Lisp sysouts (large binary images on the order of 3MB to 10MB in size), which tended to fill the incremental disk too quickly. To solve this, we reallocated disk space, moving sysouts to a directory on the backup drive itself. This kept the sysouts from being backed up incrementally. (They are still dumped monthly with the other drives and made available from a less active unit, reducing disk contention.) The removal of the sysouts from the user file system freed up large amounts of space and the user community on this server has increased in size and activity.

We moved the Xerox *Communications server* (which provides *Clearinghouse* service) to the MSOB machine room and upgraded its disk drive from 10MB to 40MB (using a

¹The last two packages are described in the *Remote Workstation Access* section of this report.

disk from our spare parts supply). We did this for several reasons. By making it our NS boot server (instead of the Xerox printer) we were able to free up memory and disk storage on the printer so that we could turn on the the *reprinting* feature of the new server software. (This allows the printing of some Interpress documents that were too complex to print before the change.) We installed file service on the new boot server so that we could store the system installation files on it. Previously the boot and installation services were split between the printer and file server. Now we are able to boot the main file server over the network from software on the Communications server rather than using the slower, less efficient floppy-based system we were limited to previously. Finally, this gave us a Clearinghouse server on the MSOB network which makes authentication activities (like logins) much faster and more reliable (independent of network/gateway irregularities).

The Xerox University Grants Program (UGP) server maintenance agreement expires at the end of this reporting year (May 31st). This program covers the file server, 15 workstations and the print server provided in the original grant. The program has been funded to extend the warranties, but the corporation headquarters hasn't given the final go-ahead. If this does not happen, we will have to cover the cost of maintenance of the file server. In this contingency, we will maintain the workstations and printer ourselves (as with our other systems).

As reported last year, the Xerox NS file server software was upgraded to support random access of data. Although we had successfully written some patches to exploit this capability, we were fortunate to be asked by Xerox to beta-test an upgrade to Xerox Lisp's NS file system interface (to make use of the new features). This code is now part of our standard system and the file server is providing services such as hashed database access and on-line manual lookups (that were not possible before).

We also beta-tested the *ROOMS* multiple virtual display window environment (developed by Xerox PARC). This environment allows the user to have any number of virtual screens at his disposal. He can switch from one to another with ease and share windows among any subset he chooses. *ROOMS* may solve the display problems of having the ONCOCIN, OPAL and OPUS systems all in the same Lisp image. For easy access in the clinic, all three systems could be left set up (on their own screens) and, using a mouse click, the user could move between them.

11.5 - Texas Instruments Explorers

The twenty Texas Instruments Explorers have enjoyed an increasing popularity as more projects have developed a need for the combination of execution speed, full Common Lisp, and sophisticated development facilities offered by the Explorer. Explorers have come into use in other parts of the national biomedical community as well, such as Ohio State University, MIT and the University of Maryland. However, the Explorer is still maturing as an AI workstation. Thus, our efforts have been directed at improving the environment of the Explorer by developing software, organizing user interest activities, and advising Texas Instruments.

Previous experience has shown that the greatest source of advancement for a particular computing environment is the user community. They are the most in touch with the deficiencies of the system, and thus uniquely positioned to address them, as well as to utilize the strengths of the system. The product developers of the system are frequently too involved in the lower levels of detail to produce general, effective solutions to problems, as well as being hampered by limited manpower resources. However, a significant amount of time and effort is required to organize this effort. This task has traditionally fallen to a user-run organization, such as DECUS or Usenix.

We have spearheaded an effort to organize an international users' group for the Explorer. The slightly misleading name for the group is NEXUS, standing for National Explorer Users' Group. The goals of this undertaking are to:

- facilitate dissemination of information by organizing meetings where presentations and discussions can be used to make little-known techniques and facilities more widely known, as well as feeding back information on needs and wants to developers,
- allow more immediate communication via electronic mailing lists, which are used for distribution of important software fixes and discussion of items of general interest, such as new software tools, or proposed changes to the system,
- publish a periodic newsletter containing usage tips, salient extracts from the electronic mailing lists, and announcements,
- and, perhaps most importantly, establish and maintain a library of public domain, user supplied software.

A steering committee has been formed and set about forming a charter for the group, consolidating membership information, arranging a meeting in conjunction with AAAI '88, and settling the legal issues involved with setting up a code library.

There are already many entries ready for the library, most of which have been developed locally. We have maintained the software tools that were produced previously by fixing bugs, making improvements, and porting to new releases. Some of these have remained essentially the same, including:

- Analog Clock
- Backgrounds
- Backup To File System
- Batch Processor
- Choice Facility Enhancements
- Deexposed Mouse
- DEFSTRUCT Type Checking
- Development Tool Consistency Enhancements
- Finger Via TCP
- General Named Structure Message Handler
- Graph Viewer
- Graphical Value Monitors
- Imagen Via TCP
- Inspector Enhancements
- Map Over Files
- Rubber Band Rectangles
- Single Window VT100
- Snapshot Windows
- Soft Keys
- Source Code Controller
- Structure Enhancements
- Symbolics 36xx to Explorer compatibility package
- Transparent Windows
- Vertically Ordered Menu Columns
- Window Manager System Menu

Many of the tools have been enhanced or newly written this year.

General Inspector	This new tool integrates the three inspectors (data, Flavor, and Class) into one tool.
Search And Replace	This new tool allows looking for a pattern in a set of files, optionally replacing occurrences of the pattern with a new string.
Source Code Debugger	This new tool allows the user to compile code with "back pointers" so that the point of execution in the source can be shown in the debugger. This tool is a non-trivial achievement.
Spelling Checker	This tool is based on TI's spelling checker, but with extensions allowing user-defined dictionaries and other more minor enhancements.
Window Debugger Enhancements	-- Additions to this tool include displaying SELF in the Locals pane even when it isn't explicitly referenced and allowing items in the Inspect pane to be modified.
Window Dragging	This new tool allows many windows to be "grabbed" by holding down the middle mouse button and dragged to new locations on the screen.
Window Icons	This new tools provides the infra-structure to allow shrinking temporarily unused windows to iconic representations.
Zmacs Enhancements	New features in this tool include: <ul style="list-style-type: none"> • Commands to manipulate tag tables • Commands to load tools and show their documentation • Commands to talk to the compiler allowing one to see the result of optimizing code or to check the args in an expression • A facility to spawn commands into a background process • Commands to load and compile systems

Of course, all of these will be provided to the user's library, and many of them have already been given to other sites, including IntelliCorp, Berkeley, ISI, University of Maryland, and Ohio State.

Third party software is less utilized, but we stay abreast of the latest releases of the expert system shell KEE. We have installed a DVI previewing system from MIT allowing TeX and LaTeX output to be viewed on the Explorer screen. We have available several other imported tools such as a Common Lisp LOOP package and Macintosh style scrolling.

We have been following the development of the Common Lisp Object System CLOS closely. Several projects have already begun using Portable Common Loops (PCL), the public domain near-implementation of CLOS. We have done a number of things to make PCL easier to use on the Explorer, including making the debugger understand PCL forms and display them legibly, making the editor understand PCL forms and manipulate them properly, constructing a PCL class inspector similar to the Flavor inspector, and fixing bugs.

In addition to producing and maintaining these software tools, we attempt to provide extensive testing and evaluation of Explorer hardware and software products in a sophisticated university research environment in order that these products work more effectively when they are distributed to the national community. This testing is critical to the development of the computing environment since the combination of concentrated in-house expertise and close links to the product developers allows a turnaround on problem fixes unavailable in the broader scope.

This year we have participated in testing TI's high-end product, the Explorer II, Releases 3.1 and 3.2, Release 2.0 of the Network File System protocol implementation, and the new microExplorer system.

The microExplorer is an add-in co-processor board for the Apple Macintosh II based on TI's VLSI Lisp chip. After initial presentations from TI we were concerned that the microExplorer software would be too limited to allow many of the applications we envisioned for the system so we had a design review session with TI's developers before the final design of the system was frozen and as a result many aspects of the system are quite a bit more amenable to future extension. We were given one of the first systems outside of TI to beta test. During this testing dozens of problems were reported and addressed. It is our hope that this work will result in high-performance, low-cost sophisticated Lisp availability, allowing greater dissemination of AI software to the national community.

In addition to specific testing and evaluation, we are constantly finding, tracking, fixing, and reporting software bugs. This year we submitted twenty-nine new bug reports on Explorer system software, twenty of which had fixes included. All of these fixes have been made available to the national community in a patch file.

As well as working on these specific problems, we have had many meetings with Texas Instruments representatives wherein we have attempted to present the needs of the national community for short- and long-term AI workstation products, covering issues including the desirability of specialized hardware, address space, programming environment versus execution speed, and the ability to utilize the AI workstation's power for routine tasks.

Of course, there is also a large number of day-to-day activities needed to keep the computing environment pleasant, including resource management (e.g., disk space allocation, printer management), assistance with file backup and magnetic tape usage, and introducing new users to the system. We have produced documents targeted at complete novice users, users of InterLisp-D machines, and users of Symbolics machines in order to facilitate user education. These documents have been used as examples at various places in the national community.

For the coming year we plan to continue development and maintenance of the software tools, perhaps adding tools such as a DARPA Internet Domain Resolver, UNIX print spooling, better IP access control, CLX and CLUE packages, automatic backup, better documentation, better software management facilities, KSL specific NEW-USER facility, a Zmacs novice mode, and an Explorer version of the TALK program, as well as aiding the growth of the users' group.

11.6 - Symbolics

Symbolics

Our work with Symbolics equipment has continued at a low level pending resolution of long-standing maintenance issues. As has been stated previously, in order for workstations to be competitive with time-shared mainframe computing resources, they must not only have a low purchase price, but must be cost-effective to

maintain. This goal is normally achieved due to the economies of scale associated with having a large number of identical parts in an installation, as well as amortizing the cost of software development over many machines. We have come to reasonable agreements with all of the workstation vendors except for Symbolics. The high costs of service, the exceptionally high price of mail-in board repair, and the lack of a reasonable self-service alternative has left us unable to justify continued support of these machines unless a workable agreement can be reached. We have been trying a self-maintenance contract which is supposed to supply parts while we do the diagnosis and replacement work. This arrangement has not worked out well due to inadequate diagnosis software and procedures and delays in getting parts from Symbolics. In the coming year, we plan to try a standard Symbolics maintenance contract.

We plan to bring up Symbolics Genera 7.2 this summer. We have installed the REFINE system as well as KEE and FORTRAN.

11.7 - SUN Workstations and Lisp Environment

Due to the high performance relative to purchase cost, Sun workstations drawn strong interest as Lisp engines. For the past year we have had three SUN 3/75 workstations in experimental use in the KSL. Because these were purchased for LISP work, we have added a 24 megabyte memory board (from Parity Systems Inc) to each of these. Also added were 70 megabyte SCSI disks. The systems have been set up to swap the large virtual memory to the local SCSI disk, rather than over the Ethernet to the server.

One of the systems is being used in the "Very Large Reusable Knowledge Base" project. A speech recognition box has been connected to another of the clients, and an interface to a Xerox InterLisp workstation running ONCOCIN developed to study the use of speech input for medical information. An evaluation of SUN workstations was made in terms of their suitability as a platform for a general physician's workstation which would support data management, analysis and display, and consultative software. For now, the SUN/UNIX environment was judged not to be competitive, in terms of cost or user interface technology, with other workstations environments for this purpose.

The vendor plans for LISP support on SUN workstations has been in a state of flux this past year. Currently their Lisp supplier is Lucid but there have been many rumors about a shift to Franz, Inc. as a replacement supplier. It appears that this uncertainty has delayed SUN's plan for improvements in their current package. However, since Franz Inc. is closely involved with CORAL in the LISP package we prefer to use with our MAC-II's, we might see a benefit in SUN making this change in suppliers.

Overall, Sun's Lisp still does not provide the programming environment power and maturity of the implementations for microprogrammed Lisp machines (see Appendix C). Sun seems to be devoting a good deal of effort to its Symbolic Programming Environment (SPE) product to alleviate this situation. We examined an early version of SPE and found it very similar to its foundation, the HyperClass system from Schlumberger. We are given to understand that SPE has made significant progress since those early versions, so we intend to re-evaluate it soon. We have been using the above mentioned HyperClass system as well as keeping new releases installed.

12 - Workstation Standards and Access

12.1 - Computing Environment Standards

In a heterogeneous computing environment, such as AI research inevitably involves, the issue of cross-system compatibility is a central one. Users of various machines want to be able to share software, as well as be able to use various machines with a minimum of overhead in learning the operating procedures and programming languages of new systems. Thus, it is crucial to specify and propagate powerful, flexible standards for various aspects of the computing environment so that it is possible to transfer both skills and information among machines.

In order to improve the inter-machine compatibility of our software, we have been encouraging all users to use the Common Lisp programming language [5], as well as pressing vendors to provide more complete and efficient implementations of this language. We have already served as beta test sites for Xerox, Texas Instruments, and Lucid Common Lisp implementations.

The Common Lisp language, however, is only a subset of the software needed for our research. Research projects need higher-level powerful facilities, such as an object-oriented programming system and sophisticated error handling. Therefore we have been supporting and following the development of the Common Lisp Object System (CLOS) via membership in the electronic discussion group, technical contributions, and porting of Portable Common Loops (PCL), a predecessor of CLOS, to the TI Explorer. We are now encouraging vendors to produce efficient implementations of the system, and users to familiarize themselves with it. We are also encouraging vendors to adopt the proposed Common Lisp error system.

Other features of the computing environment also need to be standardized to be useful on more than one machine at a time. Another of the most important of these is the keyboard and display interface, often referred to as the "window system". See the virtual graphics section (page 48) for further discussion of window systems.

There are also many other areas which could benefit greatly from standardization, including document page description languages, text and graphics representations, and more networking protocols. However, it is important that standards not be entered into hastily, as an insufficient standard can often be worse than no standard at all. We intend to continue working to develop standards for these and other computing needs as the understanding of the issues involved matures.

12.2 - Protocol Standards

Another important area of standardization has been inter-machine communication, or networking. Underlying all network I/O must be a network protocol for packet transfer between cooperating hosts. At SUMEX we have had long term experience with several such protocols; PARC Universal Packet protocol (PUP), TCP/IP, ChaosNet, and XNS/SPP. These have been used to implement higher level services such as remote booting, file transfers and access, TELNET access, electronic mail and bulletin boards, remote procedure call interfaces, remote graphics interfaces, and numerous utility services for locating network hosts, addresses, and the like. In the past we have elected to write servers for each new protocol in order to accommodate both vendor hardware and systems software requirements. This was necessary because no one protocol has been supported on all such systems.

With others in the computer science research community, we have pressed vendors to supply implementations of the DARPA standard TCP/IP communications protocols. We are pleased that the IP protocol family is now supported on all hardware and

operating system configurations currently at SUMEX. And we expect to have IP support on any new systems we purchase in the future. Similarly, IP is supported on all of our UNIX based file servers, and the SUNet gateways route all IP datagrams. There has been a great deal of effort at Stanford and SUMEX to enforce IP as a standard protocol for new software development. This was motivated by its broad acceptance and the growing number implementations throughout the networking and vendor communities. This does not imply that we will abandon the other protocols but rather, since we are seeking to have *uniformity across all vendors* with the proposed Stanford distributed environment, we are choosing to limit new implementations to the IP protocol family. We are also currently working to provide improved support for TCP/IP in our Terminal Interface Processors (TIP's), having already implemented TCP/IP routing service.

Such standardization has a price, however, in that observed network communications speeds are often higher between equipment "tuned" to individual vendor protocols. We continue our efforts to assess performance bottlenecks and to refine system implementations to achieve acceptable throughput.

13 - Network Services

A highly important aspect of the SUMEX system is effective communication within our growing distributed computing environment and with remote users. In addition to the economic arguments for terminal access, networking offers other advantages for shared computing. These include improved inter-user communications, more effective software sharing, uniform user access to multiple machines and special purpose resources, convenient file transfers, more effective backup, and co-processing between remote machines. Networks are crucial for maintaining the collaborative scientific and software contacts within the SUMEX-AIM community.

13.1 - National and Wide-Area Networks

13.1.1 - General National Networking Issues

A 2-day conference on national research computer network issues was held in Washington, DC during April 1988. It was sponsored by The National Association of State Universities and Land Grant Colleges, The EDUCOM Networking and Telecommunications Task Force, and The New York State Education and Research Network (NYSERNet) with support from Apple Computer, IBM, and NYNEX. The meeting was well organized and encompassed views from academia, federal and state government, National Academy of Science/National Research Council (NAS/NRC) and White House Office of Science and Technology Policy (OSTP) study groups, industry, and Congress. There were discussions about needs and goals, NSFNet status and plans (including the Michigan MERIT/IBM/MCI implementation of the NSFNet backbone and regional network experiences, especially NYSERNet), DARPA's viewpoint, high bandwidth network research issues, a congressional view, and many funding/legal/technical issues. The meeting had over 300 attendees representing most of the major players. No one was at the meeting from NIH. Also, NIH has apparently not been very active on the FRICC (Federal Research Internet Coordinating Committee) set up by the OSTP for interagency coordination. The FRICC includes DARPA, NSF, DoE, NASA, and HHS.

Outside of DoD activities, almost all of the justification for the national NSFNet effort has come from the supercomputer initiative. Even most of the regional networks have been set up to facilitate access to supercomputers -- e.g., NYSERNet and BARRNet. The FCCSET (Federal Coordinating Council for Science, Engineering, and